Systems Programming Assignment 2

Building the Codebook

Our code uses a hashmap to calculate the frequency of each word. The time complexity of this is dependent on the number of collisions. We used a hashmap with 1000 spaces and a linked list to resolve collisions. The average case is O(1) for most files but can reach O(N) time in the worst case.

Once we have the frequency of each word, We insert it into a max heap to create the huffman code for the word. We use a max heap to create the codebook. My code uses a max heap to create the codebook. Insertions into the heap are done in log(n) time; total time to insert all words are nlog(n) time where n is the number of unique words.

We then create the huffman tree by popping two nodes off of my max heap, and then combining them into a new node with the two nodes as children and the key to the new node being "!NONTERMINAL!". We then insert this node back into the max heap. We repeat this process until there is only one node left in the max heap. This node is the root of my huffman tree.

We use the huffman tree to print out the codes in the Huffman Codebook file.

When building the codebook recursively, we wait until we have recorded the frequency of every word in every file in my hashmap and then move on to the following steps.

Encoding a File

We read the huffman codebook file and create a hashmap with the keys as the word and the value as the code. I use a linked list to resolve collisions. The average case is O(1) and the worst case is O(N) where N is the number of unique words.

Decoding a File:

We use a hashmap to do the file decoding. I read the huffman codebook file and recreate the Huffman Tree. The huffman tree is used to read the input words. I traverse the tree by going left when hitting a 0 and right when hitting a 1, the runtime to do a lookup log(M) time and the overall runtime is Nlog(M) with N being the number of unique words in the file and M being the average length of a code.

There is no change to this process when calling the encoder recursively; it is simply applied to more files

Overall

The overall runtime of the code is O(N + Nlog(N) + Nlog(N) + Nlog(M)) where N is the number of unique words and M is the average length of a codeword. The code is bottlenecked by decoding since the coded words can get very large.